SQL PROJECT DECEMBER- 2023

1. Que-A- Data type of all columns in the "customers" table.

   Query –A

```
select column_name, data_type
 from
 scaler-dsml-sql-402706.targetsql.INFORMATION_SCHEMA.COLUMNS
```

   OUTPUT

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

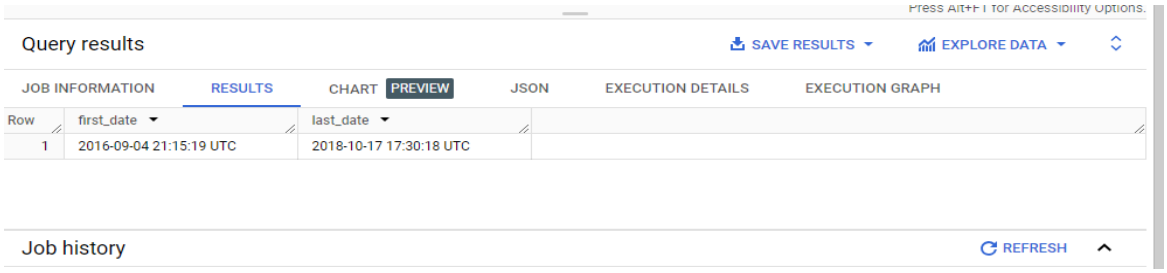
   Insights- Most of the columns in the customers table are of String Data type

1. Que B. Get the time range between which the orders were placed.

   Ans B query –

```
Select
min(order_purchase_timestamp) as first_date,
max(order_purchase_timestamp) as last_date
from `targetsql.orders`
```

   OUTPUT-



   INSIGHTS – The time range between which orders were placed is 2016-09-04 to 2018-10-17

1. **Que-1 C Count the Cities & States of customers who ordered during the given period.**

   Query 1 C count of city and states

```
select count(distinct(c.customer_city)) as customer_city,
count(distinct(c.customer_state)) as customer_state
from`targetsql.customers` c
 left join `targetsql.orders` o
 on c.customer_id = o.customer_id
 where  o.order_purchase_timestamp between (select
min(order_purchase_timestamp) from `targetsql.orders`) and (select
max(order_purchase_timestamp) from `targetsql.orders`)
```

OUTPUT

```
13  select count(distinct(c.customer_city)) as customer_city, count(distinct(c.customer_state)) as customer_state
14  from`targetsql.customers` c
15   left join `targetsql.orders` o
16   on c.customer_id = o.customer_id
17   where  o.order_purchase_timestamp between (select
18  min(order_purchase_timestamp) from `targetsql.orders`) and (select
19  max(order_purchase_timestamp) from `targetsql.orders`)
20
```

Press Alt+F1 for Accessibility Optior

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_city	customer_state
1	4119	27

INSIGHTS- there are 4119 cities and 27 states in database

Que 1C -Query order wise names of city and state-

Query 1C

```
select c.customer_city as customer_city, c.customer_state as customer_state,
count(distinct(c.customer_id)) as no_of_orders
from`targetsql.customers` c
 left join `targetsql.orders` o
 on c.customer_id = o.customer_id
 where  o.order_purchase_timestamp between (select
min(order_purchase_timestamp) from `targetsql.orders`) and (select
max(order_purchase_timestamp) from `targetsql.orders`)
group by 1,2
order by 3
```

Query results

SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_city	customer_state	no_of_orders
1	avai	SP	1
2	bodo	RN	1
3	bora	SP	1
4	caem	BA	1
5	cipo	BA	1
6	inga	PB	1
7	japi	RN	1

Results per page: 50 | 1 – 50 of 4310

Job history

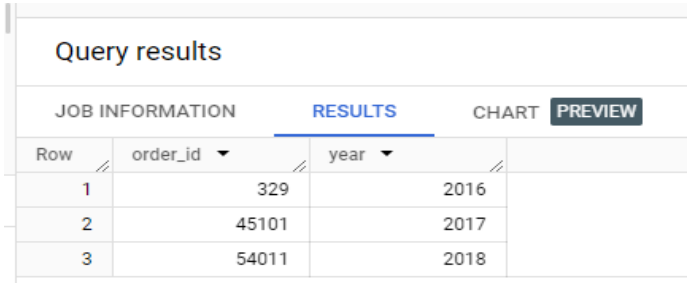
REFRESH

1. Que 2 A- Is there a growing trend in the no. of orders placed over the past years?

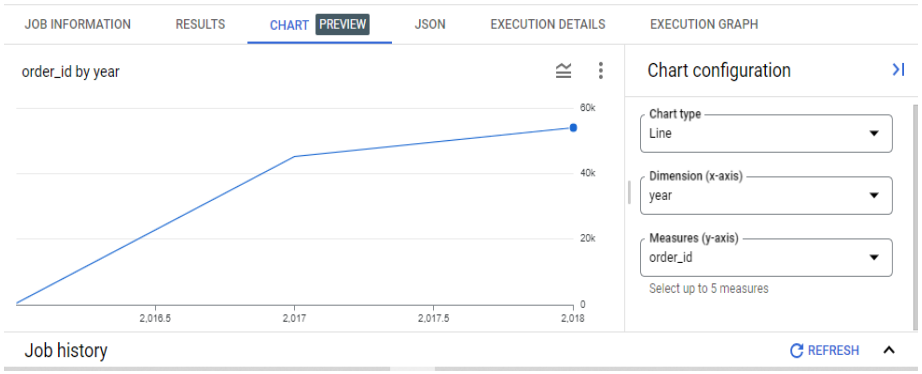
Ans 2 A Query

```
select count(order_id) as order_id,
extract(year from order_purchase_timestamp) as year
from `targetsql.orders`
group by year
order by year
```

OUTPUT



Insights- There is a generous growth among orders placed in 2017 as compared to 2016, 2018 is also showing growth in no of orders placed



Que2B. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Ans 2 B- Query

```
select count(order_id) as orders_placed,
extract(month from order_purchase_timestamp) as month
from `targetsql.orders`
group by  month
order by month
```

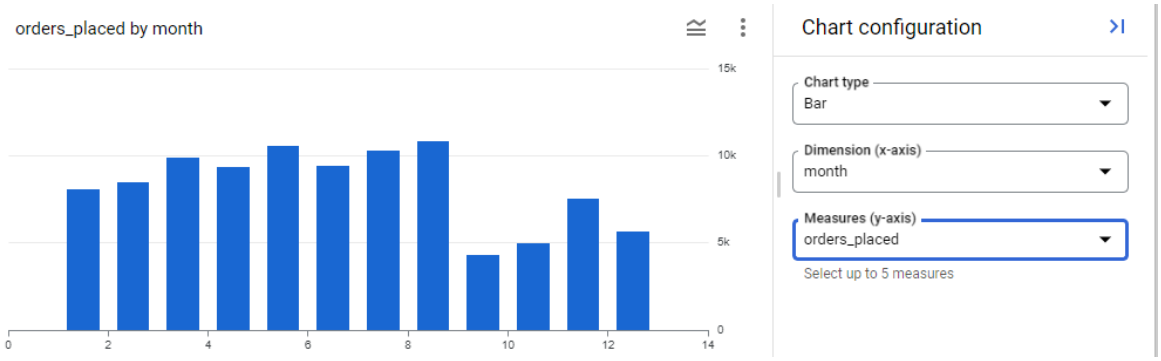
Output -

JOB INFORMATION | RESULTS | CHAR

Row	orders_placed	month
1	8069	1
2	8508	2
3	9893	3
4	9343	4
5	10573	5
6	9412	6
7	10318	7
8	10843	8
9	4305	9
10	4959	10

Insights- Highest number of orders are placed in November and lowest number of orders are placed in September

Graph-



Que 2 C – During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night

Query 2C

```
select count(order_id) as count_order,
case
when a.hours_of_orders between 0 and 6 then 'DAWN'
when a.hours_of_orders between 7 and 12 then 'Morning'
when a.hours_of_orders between 13 and 18 then 'Afternoon'
else 'night'
End as Time_of_day
from
(select *,
extract(hour from order_purchase_timestamp) as hours_of_orders
from `targetsql.orders`)a
group by Time_of_day
```

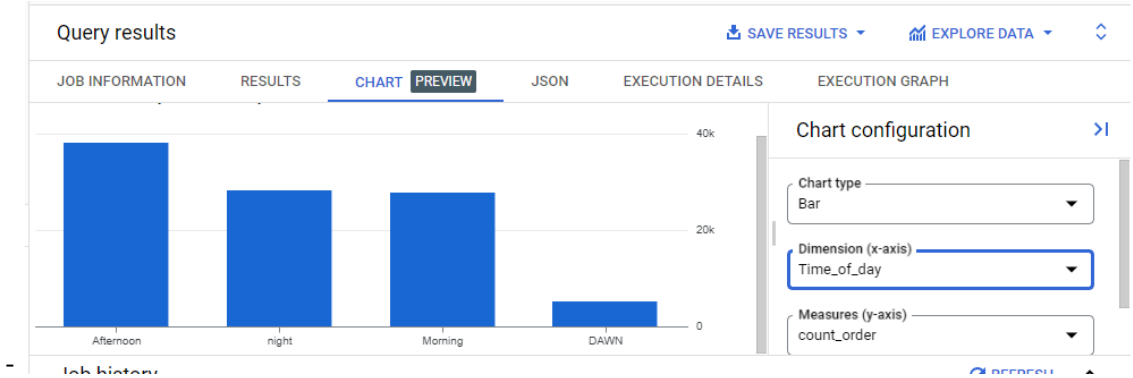
Output-

Query results

JOB INFORMATION | RESULTS | CHART | PREV

Row	count_order	Time_of_day
1	38135	Afternoon
2	28331	night
3	27733	Morning
4	5242	DAWN

Graph



Insights- Highest orders placed in afternoon and lowest no of orders placed in Dawn

1. Que 3 A- Get the month on month no. of orders placed in each state

Ans 3 A Query -

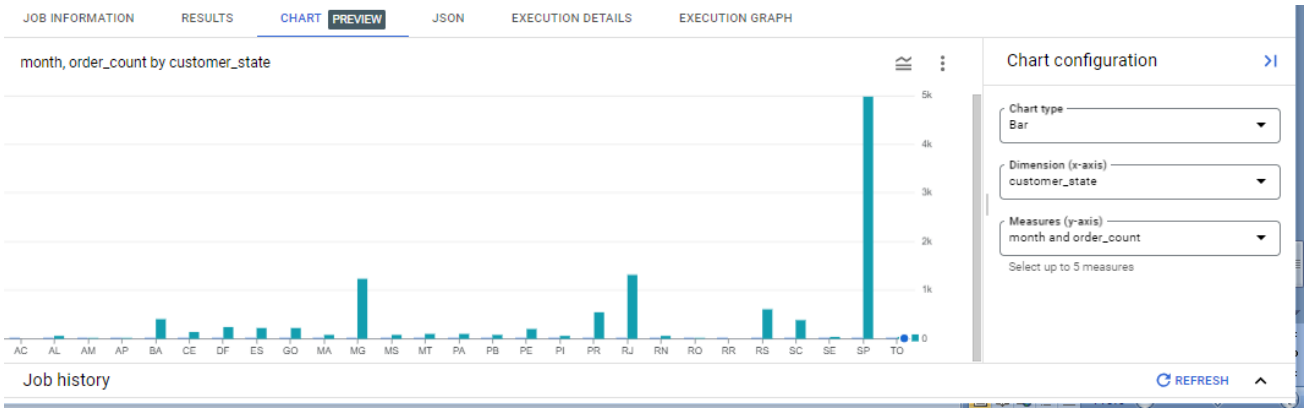
```
SELECT
  c.customer_state,
  EXTRACT(month FROM o.order_purchase_timestamp) AS month,
  COUNT(o.order_purchase_timestamp) AS order_count
FROM
  `targetsql.orders` o
JOIN
 `targetsql.customers` c
ON
  o.customer_id = c.customer_id
GROUP BY
  c.customer_state, month
ORDER BY
  c.customer_state, month
```

OUTPUT

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS

Row	customer_state	month	order_count
19	AL	7	40
20	AL	8	34
21	AL	9	20
22	AL	10	30
23	AL	11	26
24	AL	12	14
25	AM	1	12
26	AM	2	16
27	AM	3	14
28	AM	4	19

GRAPH



INSIGHTS- Highest number of orders are placed in SP. More details are given in output

Que 3 B How are the customers distributed across all the states?

Ans 3B Query-

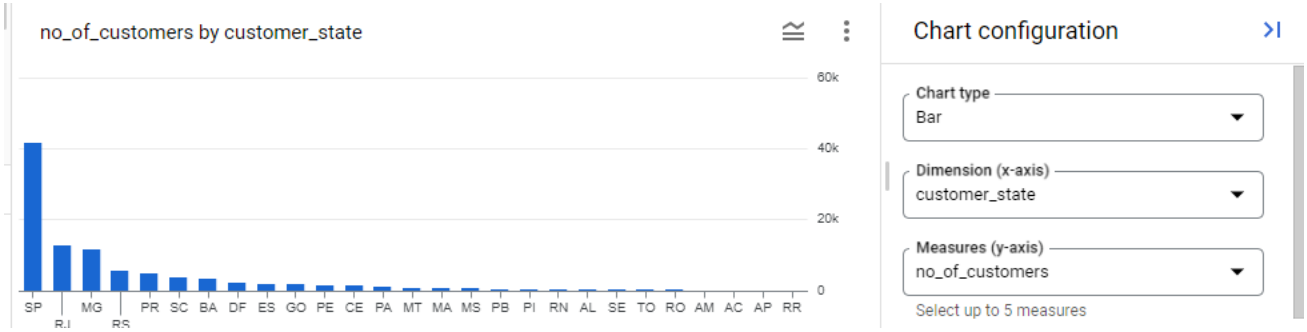
```
SELECT
customer_state,
  COUNT(customer_id) AS no_of_customers
FROM
  `targetsql.customers`
GROUP BY  customer_state
ORDER BY no_of_customers DESC;
```

Output

JOB INFORMATION | RESULTS | CHART PREVIEW

Row	customer_state	no_of_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140

Graph-



Insights- Highest orders placed in SP-4176 orders and lowest in RR- 46 orders

Que -4A Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
Query 4 A-

```
with final as
(
select extract(year from o.order_purchase_timestamp) as order_year,
extract(month from o.order_purchase_timestamp) as order_month,
sum(p.payment_value) as total
from `targetsql.orders`o
  join `targetsql.payments` p
  on o.order_id = p.order_id
  where  extract(year from o.order_purchase_timestamp) IN (2017,2018)
  and extract(month from o.order_purchase_timestamp) BETWEEN 1 AND 8
   group by order_year,order_month)

  select (sum(case when order_year = 2018 then total END)
  -sum(case when order_year = 2017 then total END) ) /
  sum(case when order_year = 2017 then total END)*100 as Percent_increase
  from final
```

OUTPUT-

JOB INFORMATION | RESULTS | CH

Row	Percent_increase
1	136.9768716466...

INSIGHTS – THERE IS 136.97 % GROWTH IN COST OF ORDERS FROM 2017 TO 2018

1. Que 4B- Calculate the Total & Average value of order price for each state.

   Query-

```
SELECT
  c.customer_state,
  EXTRACT(month FROM o.order_purchase_timestamp) AS month,
  COUNT(o.order_purchase_timestamp) AS order_count
FROM
  `targetsql.orders` o
JOIN
 `targetsql.customers` c
ON
  o.customer_id = c.customer_id
GROUP BY
  c.customer_state, month
ORDER BY
  c.customer_state, month
```

OUTPUT-

Query results

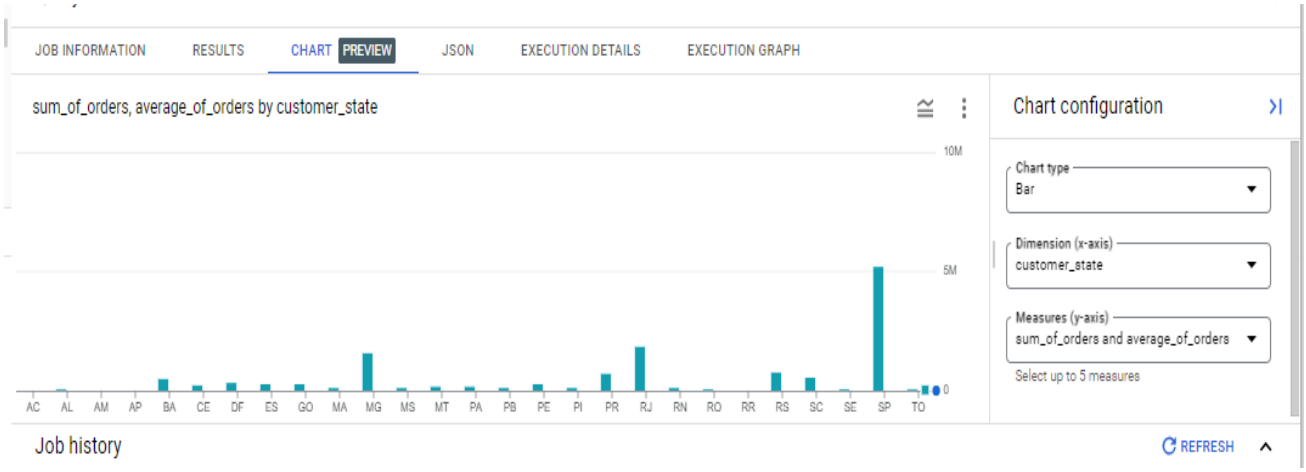
JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DET

Row	customer_state	sum_of_orders	average_of_orders
1	AC	15982.9499999...	173.727717391...
2	AL	80314.8100000...	180.889211711...
3	AM	22356.8400000...	135.495999999...
4	AP	13474.2999999...	164.320731707...
5	BA	511349.990000...	134.601208212...
6	CE	227254.709999...	153.758261163...
7	DF	302603.939999...	125.770548628...
8	ES	275037.309999...	121.913701241...
9	GO	294591.949999...	126.271731675...
10	MA	119648.219999...	145.204150485...
11	MG	1585308.02999...	120.748574148...

oad more

INSIGHTS- The output displays sum of order for each state and average of orders for each state. For details please check the output.

GRAPH-



1. **QUE-4C** Calculate the Total & Average value of order freight for each state.

QUERY -

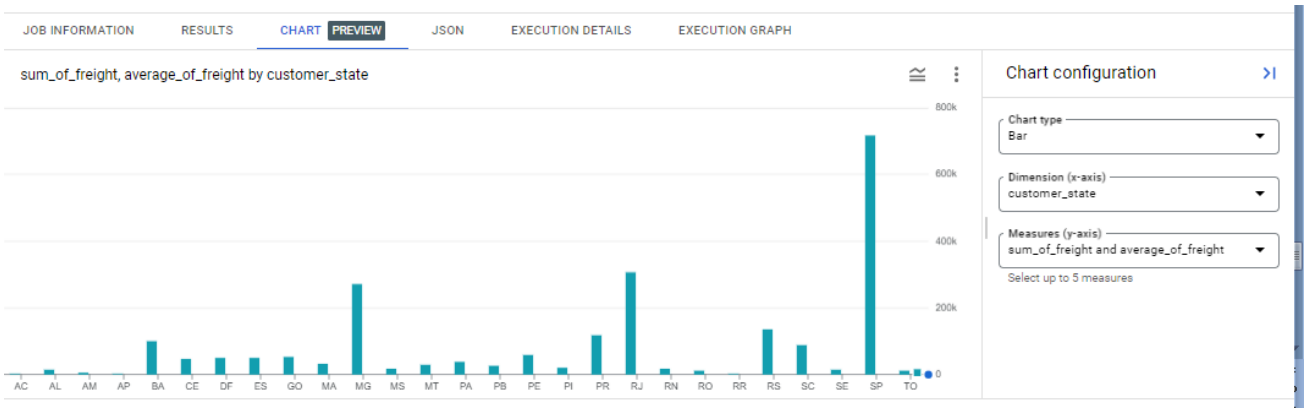
```
select c.customer_state,
sum(oi.freight_value) as sum_of_freight,
avg(oi.freight_value) as average_of_freight
from `targetsql.customers` c
join `targetsql.orders` o
on c.customer_id = o.customer_id
join
`targetsql.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state
order by c.customer_state
```

OUTPUT-

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXE

Row	customer_state	sum_of_freight	average_of_freight
1	AC	3686.74999999...	40.0733695652...
2	AL	15914.5899999...	35.8436711711...
3	AM	5478.88999999...	33.2053939393...
4	AP	2788.50000000...	34.0060975609...
5	BA	100156.679999...	26.3639589365...
6	CE	48351.5899999...	32.7142016238...
7	DF	50625.4999999...	21.0413549459...
8	ES	49764.5999999...	22.0587765957...
9	GO	53114.9799999...	22.7668152593...
10	MA	31523.7700000...	38.2570024271...
11	MG	270853.460000...	20.6301668063...
12	MS	19144.0300000...	23.3748840048...

GRAPH



Insights- SP has highest total and average freight charges

Q-5 A- Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order. Do this in a single query.

5A query –

```
select order_id, timestamp_diff(order_delivered_customer_date, order_purchase_timestamp,
DAY) as time_to_deliver,
timestamp_diff(order_estimated_delivery_date, order_delivered_customer_date, DAY) as
diff_estimated_delivery
from `targetsql.orders`
where timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, DAY) IS
NOT NULL
```

```
order by time_to_deliver, diff_estimated_delivery
```

OUTPUT-

Row	order_id	time_to_deliver	diff_estimated_deliv
1	d5fbeedc85190ba88580d6f82...	0	7
2	79e324907160caea526fd8b94...	0	8
3	e65f1eeee1f52024ad1dcd034...	0	9
4	b70a8d75313560b4acf607739...	0	9
5	1d893dd7ca5f77ebf5f59f0d20...	0	10
6	d3ca7b82c922817b06e5ca211...	0	11
7	f3c6775ba3d2d9fe2826f93b71...	0	11
8	21a8ffca665bc7a1087d31751...	0	11
9	f349cdb62f69c3fae5c4d7d3f3...	0	12
10	38c1e3d4ed6a13cd0cf612d4c...	0	16
11	434cecee7d1a65fc65358a632...	0	19
12	bb5a519e352b45b714192a02f...	0	25
13	8339b608be0d84fca9d8da68b...	0	27
14	da8831dfbb89ea6b128840224...	1	0

Insights- the output shows delivery time and difference between estimated and actual delivery for each order id

1. **5B QUE-** Find out the top 5 states with the highest & lowest average freight value.
2.

Query 5B-

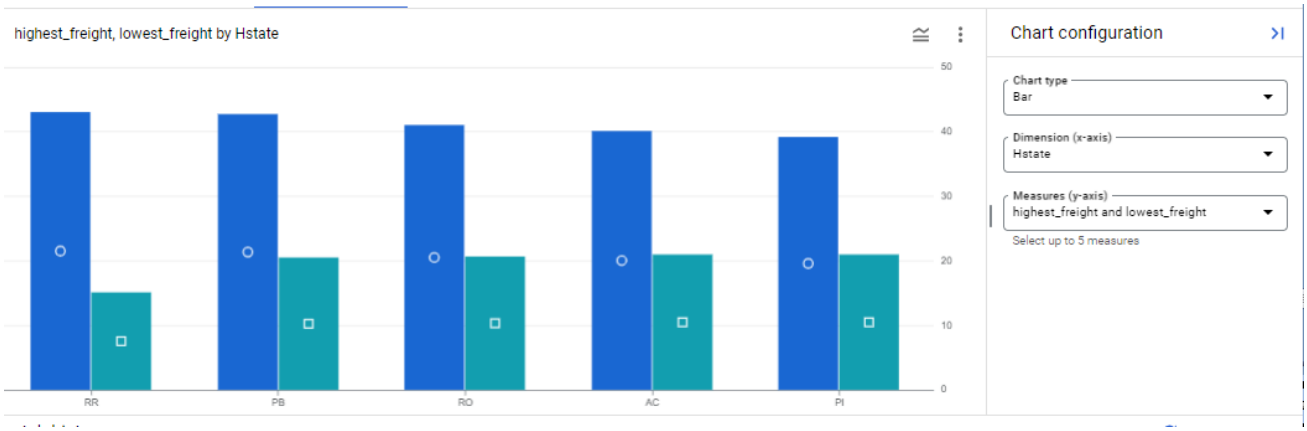
```
select a.customer_state as Hstate, a.average_freight_value as
highest_freight,b.customer_state as Lstate, b.average_freight_value as lowest_freight
from
(SELECT c.customer_state, avg(oi.freight_value) as average_freight_value,
row_number() over (order by avg(oi.freight_value) desc) as f_value,

from `targetsql.customers` c
join `targetsql.orders` o
on c.customer_id = o.customer_id
join
`targetsql.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state) a
join
(SELECT c.customer_state, avg(oi.freight_value) as average_freight_value,
row_number() over (order by avg(oi.freight_value) ) as f_value
from `targetsql.customers` c
join `targetsql.orders` o
on c.customer_id = o.customer_id
join
`targetsql.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state) b
on  a.f_value = b.f_value
limit 5
```

OUTPUT

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION DETAILS | EXE

Row	Hstate	highest_freight	Lstate	lowest_freight
1	RR	42.9844230769...	SP	15.1472753904...
2	PB	42.7238039867...	PR	20.5316515679...
3	RO	41.0697122302...	MG	20.6301668063...
4	AC	40.0733695652...	RJ	20.9609239316...
5	PI	39.1479704797...	DF	21.0413549459...

GRAPH-



Insights- the highest freight value is for RR and lowest greight value is for SP

1. Que5C- Find out the top 5 states with the highest & lowest average delivery time.
2.

Query –

```
select a.customer_state as Hstate, a.time_to_deliver as highest_time,b.customer_state as Lstate,
 b.time_to_deliver as lowest_time
from
(SELECT c.customer_state,timestamp_diff(order_delivered_customer_date, order_purchase_timestamp,
DAY) as time_to_deliver,
row_number() over (order by avg( timestamp_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY)) desc) as d_value,

from `targetsql.customers` c
join `targetsql.orders` o
on c.customer_id = o.customer_id
join
`targetsql.order_items` oi
on o.order_id = oi.order_id
group by c.customer_state,time_to_deliver ) a
join
(SELECT c.customer_state, timestamp_diff(order_delivered_customer_date, order_purchase_timestamp,
DAY) as time_to_deliver,
row_number() over (order by avg(timestamp_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY)) ) as d_value
from `targetsql.customers` c
```

```
join `targetsql.orders` o
on c.customer_id = o.customer_id

join
`targetsql.order_items` oi
on o.order_id = oi.order_id
where timestamp_diff(order_delivered_customer_date, order_purchase_timestamp, DAY) IS NOT NULL
group by c.customer_state, time_to_deliver) b
on  a.d_value = b.d_value

limit 5
```

OUTPUT-

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EXECUTION DETAILS | EXEC

Row	Hstate	highest_time	Lstate	lowest_time
1	ES	209	SP	0
2	RJ	208	RJ	0
3	PA	195	BA	0
4	PI	194	SP	1
5	SE	194	RJ	1

Insights – The output shows highest time as 209 days for ES and lowest time as 0 days for SP, RJ,and BA

Que -5D Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

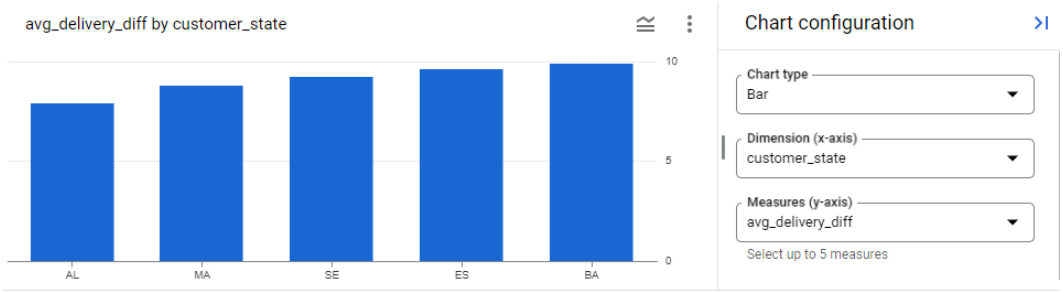
Query 5D -

```
with DeliveryTimeDiff as(
  select c.customer_state,
ROUND(avg(timestamp_diff(order_estimated_delivery_date,order_delivered_customer_
date, DAY)),1) as avg_delivery_diff
  from `targetsql.customers` c
  join `targetsql.orders` o
  on c.customer_id = o.customer_id
  where order_delivered_customer_date is not null
  group by c.customer_state
)

select customer_state, avg_delivery_diff
from DeliveryTimeDiff
order by avg_delivery_diff ASC
limit 5
```

OUTPUT -

Row	customer_state	avg_delivery_diff
1	AL	7.9
2	MA	8.8
3	SE	9.2
4	ES	9.6
5	BA	9.9

GRAPH-



INSIGHTS- State AL has fastest delivery , other top 4 are displayed in output and graph

1. Que -6A Find the month on month no. of orders placed using different payment types.
Query 6 A-

```
select extract (year from o.order_purchase_timestamp) as order_year,
extract (month from o.order_purchase_timestamp) as order_month,
p.payment_type, count(p.order_id) as order_count,
from `targetsql.orders`o
join `targetsql.payments` p
 on o.order_id = p.order_id
 group by order_year, order_month, payment_type
order by  order_year, order_month, payment_type
```

OUTPUT

Row	order_year	order_month	payment_type	order_count
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61
11	2017	2	UPI	398
12	2017	2	credit_card	1356
13	2017	2	debit_card	13

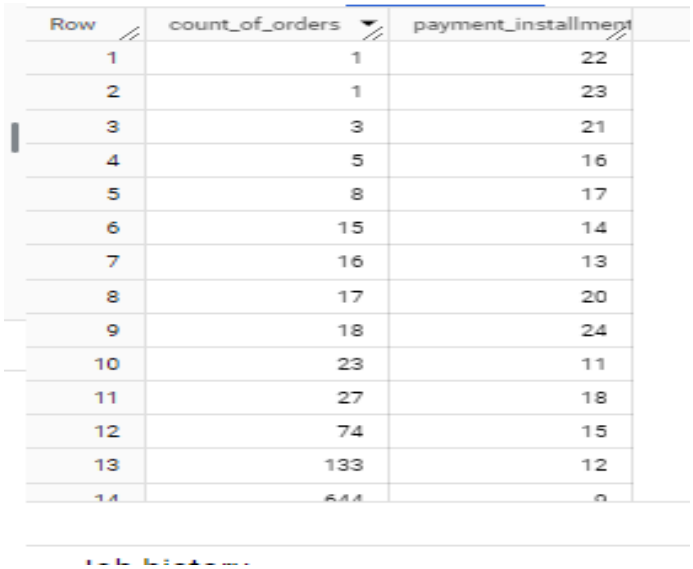
Insights – THE output shows no of orders placed using different payment types in each year and each month.
Credit card is the highly used payment type .

1. **Que 6 B** Find the no. of orders placed on the basis of the payment installments that have been paid.

```
Query 6B select count(o.order_id) as count_of_orders, p.payment_installments
from `targetsql.orders` o left join
`targetsql.payments` p
on o.order_id = p.order_id
where payment_installments>= 1
```

```
group by p.payment_installments
order by count_of_orders, p.payment_installments
```

OUTPUT



Row	count_of_orders	payment_installment
1	1	22
2	1	23
3	3	21
4	5	16
5	8	17
6	15	14
7	16	13
8	17	20
9	18	24
10	23	11
11	27	18
12	74	15
13	133	12
14	644	0

Insights-The output shows no of orders placed on the basis of the payment values based on payment in installments